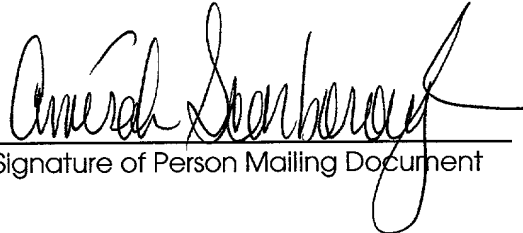


CERTIFICATE OF MAILING UNDER 37 CFR§ 1.10

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail in an envelope addressed to: Assistant Commissioner of Patents, Washington, DC 20231 on **November 7, 2001**

EXPRESS MAIL LABEL: **EL 888549940 US**

Amirah Scarborough
Name of Person Mailing Document


Signature of Person Mailing Document

**SYSTEM AND METHOD FOR PHYSICAL MEMORY ALLOCATION IN
ADVANCED OPERATING SYSTEMS**

BACKGROUND OF THE INVENTION

1. Technical Field:

The present invention relates generally to data processing systems and in particular to memory allocation in data processing systems. Still more particularly, the present invention relates to a method and system for specifying a particular physical memory location to allocate to an application running on a data processing system.

2. Description of The Related Art:

A standard data processing system comprises one or more central processing units (CPU), one or more levels of caches, one or more memory, and input/output (I/O) mechanisms all interconnected via an interconnect. Traditionally, the interconnects utilized consisted primarily of a system bus and an I/O bus. In newer processing systems, however, particularly those with large numbers of CPUs and distributed memory, a switch is often utilized as the interconnecting mechanism. In current systems also, the configuration of the data processing system may involve linking of separate systems (processor groups with associated memory, etc.) in a node-configured distributed system.

For example, non-uniform memory access (NUMA) systems comprise this node configuration.

In addition to the major hardware components, a major software (or firmware) component of a data processing system is an Operating System (OS). The OS provides an environment for the execution of programs by providing specific services to the programs including loading the programs into memory and running the program. The OS also provides resource allocation when there are multiple users or jobs running simultaneously. One of the more widely utilized operating systems is Windows (a trademark of Microsoft Corporation).

In order for an application to operate correctly, it must interact in a somewhat predictable way with the OS. Thus, application developers typically take their applications through a series of operational runs before placing the application on the market. Also, applications running on the computer system utilizes a portion of available memory during operation. Thus, in addition to consideration of the OS, consideration is often also given to the allocation and utilization of memory by the application. The OS is responsible for allocating memory and deallocating memory to the various applications (via memory controller) and thus utilization of memory by the application is dependent on the OS' method of allocating and deallocating memory.

Memory comprises a large array of words or bytes, each with its own address. During operation of an application, most processing systems allow the application's process(es) to reside in any part of physical memory. The portion of memory allocated may be selected at random (although the present invention allows selection of specific areas of memory). Basically, the key allocation step involves mapping of symbolic program address (i.e., a logical or virtual address) to actual physical address (in the memory). The virtual memory address is placed within the memory address register and

a run time mapping from virtual to physical address is completed by a memory management unit (MMU).

When all of the memory may be allocated, performing the memory allocation is accomplished relatively easily. However, if specific areas of the memory need to be tested (or allocated), then the allocation and any testing becomes fairly difficult. With traditional data processing system configuration, i.e., systems with a single, congruent memory, allocating and/or testing the memory during program execution is easily accomplished. The OS is only able to allocate blocks within the single memory of the system.

In today's computer systems, the system memory is managed by the operating system and is allocated to different software applications as needed. Use of virtual memory is a technique by which a relatively smaller amount of physical memory can be made to seem larger and shareable among many processes. Each software application therefore deals with "effective" addresses in a virtual memory space, which allow the application to read, write, and execute when required, without ever being concerned with the actual locations, in physical or disk memory, where the operations are taking place. The physical address space is considered as a contiguous space. Each physical address corresponds to a storage location in a memory bank within a system memory. The application relies on the OS to perform the mapping from the effective address to a physical address and the allocation of the physical address to the requesting process.

In a typical data processing system, information is typically loaded in a system memory wherever free space is available. Thus, a virtual address of a block of information usually does not reflect the physical address (or actual address) in the system memory in which the information is actually stored. Thus, present applications that run under the Windows OS cannot specify/request a specific physical address when polling

for allocated memory from the OS. Likewise, the OS does not provide any detail of the allocated memory (i.e., physical address(es) to the applications.

5 Newer operating systems, such as Windows 2000 are able to address up to 64G (gigabytes) of physical memory, which, as explained above, may be any memory block across the system. The ability to control the exact memory location allocated to an application during testing and/or execution in order to properly predict memory allocation thus becomes extremely important.

10 European patent EP 0 851 353 A1 describes a method of allocating memory space in a main memory of a computer system to a unified memory architecture device and also describes how physical memory can be organized. This is accomplished by getting a linear address range, and mapping physical memory. Also, Microsoft Windows 2000 provides separate functions to allocate and/or deallocate an amount of physical memory, allocate a linear address range, and map the physical memory to the linear address range in an application program. Microsoft Windows 2000 further provides a driver function to convert a linear address to a physical address. In Microsoft's implementation, when allocating physical memory, the physical memory is locked down and cannot be swapped/exchanged.

20 Also, when testing memory allocation in a multi-node configured system such as "NUMA", where memory resources can be located across two different processing systems, for performance reasons, there is a high probability that the Windows OS (e.g., Whistler and beyond) running on a first system will try to allocate the memory on the same system. However, the OS' memory allocation functions in the multi-node system is not easily predicted because the OS may occasionally allocate the memory of a second system during processing to a process running on a first data processing system.

25

Thus, for example, a process running on the CPU in the first system that comprises a memory may selectively allocate the memory resources located on the first system, the memory resources on the second system, or both memory resources, and there is currently no way for the application developer to specify which specific physical block of memory to allocate to the application. Because of this occurrence, which is currently unpredictable, the present invention recognizes that predicting memory allocation and usage by a process running on both a single node and multi-node configured data processing system is sometimes desirable.

Based on the foregoing, the present invention recognizes that it would be desirable to provide application directed memory access. A method, system and program product by which application developers can direct the operating system which address blocks of memory to allocate to the application when executing on the data processing system would be a welcomed improvement. It would be further desirable to extend these memory allocation features to a multi-node configured system with distributed memory across different systems. These and other benefits are provided by the invention described herein.

SUMMARY OF THE INVENTION

5 In accordance with a preferred embodiment of the present invention, disclosed is a method and system for allocating pre-selected physical memory locations to an application executing on a data processing system. A kernel mode driver and memory allocation subroutines are provided. The memory allocation subroutines, interacting with the programming interfaces of the OS allocates and locks down blocks of memory. The memory allocation subroutines then deallocates the memory blocks based on whether or not the memory blocks fall within the pre-selected range of physical memory locations. The physical memory locations of the blocks locked down are discovered using the driver. The driver takes the virtual address of the specified memory locations and returns with a corresponding physical address.

10 This process involves the use of AWE APIs of the OS, which allows the physical memory to be locked down. The memory allocation subroutines provides functions that allow the program developer to specify the number of physical pages to allocate and a range of physical addresses. The memory allocation subroutines also comprises the algorithm(s), that allocates the physical memory within the selected range.

15 The preferred embodiment of the invention is completed with Windows 2000 OS and its provided functional features. The invention expands the functionality of the OS to enable different allocation and organization of physical memory besides the regular physical memory allocation functions provided. In one embodiment, specific formulas are utilized to determine the maximum amount of memory to allocate without grabbing too much memory, and memory is allocated in chunks when requested by the application.

20 One embodiment of the invention extends the memory allocation features to a multi-node computer system having more than one memory. Accordingly, this

embodiment allows pre-selection of specific memory blocks across nodes as well as in local memory. This embodiment, is particularly useful for testing memory allocation across nodes in such a multi-node configured system.

5

In the preferred embodiment, the memory allocation routines are a component part of the application housed in memory. In another embodiment, these routines are programmed within the software of the data processing system as a dynamic link library file. In a third embodiment, the memory allocation subroutines, particularly the lock down and allocation/deallocation processes are completed by a driver that are provided the necessary functional software and hardware elements to complete the functions entailed.

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives, and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a block diagram illustrating the core components and peripheral devices of a data processing system with which the preferred embodiment of the invention may be implemented;

Figure 2 is a block diagram of relevant software and hardware components of the data processing system of **Figure 1** in accordance with a preferred embodiment of the invention;

Figure 3A is a block diagram illustrating the core components and peripheral devices of a multi-node data processing system with multiple memory components within which one embodiment of the invention may be implemented;

Figure 3B is a block diagram of the memory components of **Figure 3A** and associated physical addresses in accordance with one embodiment of the invention;

Figure 3C is another representation of memory components of **Figure 3B** as seen from the memory controller, driver, and OS in accordance with one implementation of the present invention in a multi-node configured computer system.

Figure 4 is a flow chart of the process by which specific memory blocks are allocated to an application utilizing the features of a preferred embodiment of the present invention; and

5

Figure 5 is a flow chart of the process of **Figure 4**, including several additional processing steps in accordance with an alternate embodiment of the present invention.

FIG. 4

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

5 The present invention provides a method, system, and program product by which an application developer may specify a physical memory location to allocate to the application's processes when the application is executing on a data processing system. The invention finds applicability in most data processing systems (with specific operating system (OS) functionality), however the preferred embodiment will be described with reference to a single node data processing system (i.e., having a processor or processor group coupled to a single contiguous memory as illustrated in **Figure 1** and described below. An alternate embodiment of the invention is implemented in a multi-node data processing system comprising two separate processing systems having separate CPUs and separate memory components that are connected via a node as illustrated in **Figure 3A**.

10
15
20
25 With reference now to **Figure 1**, there is illustrated a data processing system in which a preferred embodiment of the invention may be implemented. Data processing system **100** comprises several major components, including central processing unit (CPU) (or processor) **103**, I/O devices **105** and memory **107**. CPU **103**, I/O devices **105** and memory **107** (e.g., DIMM) are interconnected via a system bus **109**. In order to properly execute related logic functions, I/O devices **105** and memory **107** both have respective controllers. Thus I/O controller **106** manages input and output processes between the I/O devices **105** and the CPU **103** (or memory **107**). Likewise, memory controller **108** manages operations involving memory access and allocation/deallocation of memory **107**.

Although, the illustrated embodiment depicts only basic components of a data processing system interconnected via a system bus, it is understood that data processing system **100** may be more complex. Data processing system **100** may comprise additional

processing systems, with each processing system having additional components (or similar components with a different configuration). Component parts of a data processing system may also be interconnected via other methods, such as a switch, for example. Presentation of a particular data processing system configuration is therefore not limiting on the scope of the invention.

Illustrated within memory **107** is operating system (OS) **113** and application **115**. Both OS **113** and application **115** are functional, software encoded components. The OS **113** resides in memory **107** on a permanent basis. Application **115** refers to processes being currently run on the data processing system **100** (by the processor **103**) and whose processes temporarily utilize portions of memory. That is, application processes are temporarily loaded into memory **107** when running. OS **113** is responsible for allocation and deallocation of memory space as needed by application **115**. System calls provide the interface between a process of the application and the OS. According to the preferred embodiment, these system calls include encoded calls for allocation of specific memory blocks to the application process, as described below. The illustrative embodiment provided herein utilizes Windows 2000 as the OS and takes advantage of the functionality of Windows 2000, specifically its AWE application programming interfaces (APIs) as described below.

The invention requires utilization of a physical memory access procedure that is available to the processing system, however, key to the invention is the utilization of an address translation device that maps virtual addresses to actual physical addresses and returns the physical addresses. The process by which this is accomplished is described below but is not meant to be limiting on the invention. Address translation is the mechanism by which effective addresses generated by the CPU to access virtual memory are translated into real (physical) memory addresses. Several types of address translation operations may be utilized. In the preferred embodiment, a kernel-mode driver (e.g., the

Getphysaddr driver) is utilized to complete the translation functions (i.e., virtual to physical address conversion) described in more detail below required by the other processes of the invention.

5 The present invention enables executing application processes to specify pre-selected physical memory locations to the OS when requesting an allocation of memory for that process. Implementation of the invention requires two major components: (1) a kernel-mode driver; and (2) memory allocation subroutines. In the preferred embodiment, a Getphysaddr driver is utilized. The preferred embodiment of the invention makes use of the driver function written by Microsoft, which obtains the physical address from the linear or virtual address. In the preferred embodiment, the invention "wraps" the driver function so that the physical address can be sent to the application. Thus, the driver takes a virtual address provided by the application and returns the corresponding physical address. The memory allocation subroutines allocate and deallocate memory and uses the driver appropriately to get the required range of physical addresses. The allocation and deallocation is done through the AWE APIs which are provided by the OS and allows physical memory to be locked down.

10
15
20 One alternate implementation includes the functional aspects of the memory allocation routine as coded operations of the driver. Thus, the driver is provided the ability to allocate physical memory within a certain address range once the actual physical addresses have been determined. However, this requires further logic implementation and current OS operations are sufficient for providing the allocation features in conjunction with the MAS.

25 Thus, the implementation of the invention requires creation of a software-enabled memory allocation routine(s), which is able to link with the AWE APIs of the OS to direct specific allocation and deallocation of selected memory blocks. **Figure 2**

illustrates the major components of the invention (both hardware and software). The interaction of the various components to effect the features of the invention are described following the description of **Figure 2**. As shown, **Figure 2** includes major software components **201** and hardware components **211**. Software components comprise an application **115**, which "sits" on top of the OS **113**. As described above, with respect to **Figure 1**, these software components actually reside within memory, although application **115** may technically exist on another storage medium, with only relevant processes being temporarily stored within memory **107**. Within application **115**, are a number of memory allocation subroutines **217** (hereinafter referred to as MAS **217**). Within OS **113** are a number of AWE APIs **219**, by which the application and specifically MAS **217** is able to interact with the OS **113** to complete the processing required. According to the preferred embodiment, the MAS **217** are encoded within a dynamic link library (dll) file.

Hardware (firmware) components include CPU **103**, kernel mode driver **221**, memory controller **108**, and memory **107**. For illustrative purposes, memory **107** is assumed to contain physical memory blocks with addresses 00000-31999. Thus, an application developer may specify a range, e.g., 10000-20000 and a size, e.g., 4K or 4096 bytes for a 32-bit Intel processor (or 8K for an Alpha processor), for allocation, and the memory allocation subroutine then allocates 4096 bytes of memory in the address range of 10000 to 20000, as will become clear later. Interconnection between the various components are illustrated by arrows. The OS and Application are "connected to" (and/or provide processes for execution on) the CPU **103**. The OS is further "connected to" the kernel mode driver **221** so as to trigger the translation functions by which physical addresses are generated and returned to the OS's AWE APIs.

In the preferred embodiment, pre-selection of the specific physical memory addresses for allocation to the application's processes are completed by the application

5 developer utilizing an interactive component of the MAS 217. The memory allocation subroutines are packaged into a MAS dynamic link library (MAS.dll). These allocation subroutines provide certain functions, which allows the application developers to specify several parameter values. In the preferred embodiment, the programmer is prompted to enter/select parameter values including: (1) the number of physical pages to allocate, (2) a desired address range of physical memory, and (3) a specific algorithm from among several available algorithms that may be utilized to allocate the physical memory within the range. The functions utilize these inputs when the application is being executed on the data processing system to return a memory structure, which is compatible for use with the AWE application programming interfaces (API)s.

10 According to the preferred embodiment, which implements a first (default) algorithm, a series of repetitive/sequenced steps are utilized to complete the selective allocation and subsequent deallocation processes. The initial step requires that either all or some of the non-swappable physical memory is allocated and locked by MAS via the OS. This allocation and lock down of portions of or all of the memory is enabled by the application functions provided by the OS (e.g., Microsoft 2000) in conjunction with the kernel mode driver. References to "all" memory may include certain limitations. For example, "all" may be limited by the maximum amount of physical memory that the OS will allocate to a single program/process at the particular time. Also, there could be other artificial limits on what "all" could be, such as when trying to allocate only the available physical memory at the time rather than all of the physical memory in the system. Finally, memory may not be able to be allocated at once, and could be allocated in smaller amounts. Accordingly, in one of the available algorithms, The smaller block of memory is then processed (i.e., a determination made about whether the addresses are within the pre-selected range) before a next block can be allocated and processed.

Once the allocation of all memory is completed, the physical memory allocated is analyzed, and the physical pages that fit into the address range desired are identified. The pages that fit into the desired range are marked or moved so that they are kept by (or allocated to) the application. In other embodiments, additional processing may occur. For example, all the physical pages allocated may be sorted by address, and then the pages may be checked to see if they fit in the range. Several variations of the algorithm and thus the type of analyses desired are provided, and any one may be selected by the application developer. According to the preferred embodiment, a release (deallocation) of all the physical memory that is not within the desired physical range of the memory is completed utilizing the application functions provided by the OS.

An algorithm (pseudo-code) for completing one embodiment of the features of the invention is provided below and further explained in the process flow of **Figure 4**.

The pseudo code for the basic algorithm is as follows:

Begin

Input Parameters:

NumberOfPagesRequested

LowerBoundOfRange

UpperBoundOfRange

// Set the Maximum amount of memory to allocate

MaximumMemoryToAllocate = AmountOfPhysicalMemoryInTheSystem /
PageSize;

// Loop until the memory runs out, the number of pages requested has been
satisfied, or the maximum memory to allocate has been reached

while((MaximumMemoryToAllocate has not been allocated and
CurrentInRangePages < NumberOfPagesRequested and

PhysicalAllocate() is still returning addresses)

{

// Allocate the physical memory putting the allocated pages into the

5 PageArray

physicalpagesallocated = PhysicalAllocate(NumberOfPagesRequested,

PageArray)

// Loop through all the pages allocated and see if any fit in the range

10 for(x=0; x < physicalpagesallocated && CurrentInRangePages <
NumberOfPagesRequested; x++)

{

// Check to see if the page fits within the upper and lower bounds

15 if(PageArray[x] physical address <= UpperBoundOfRange &&
PageArray[x] physical address >= LowerBoundOfRange)

{

// Save the page for future use

save page in CurrentInRangePages array

CurrentInRangePages++;

20 }

else

{

// Page is unneeded, mark it, but don't deallocate it just yet
mark page for physical deallocation

25 }

}

}

Deallocate all pages marked for deallocation and pages not upper bound/lower bound tested

Return the CurrentInRangePages array and CurrentInRangePages

End

Figure 4 illustrates the flow of the general process for performing the specific memory allocation according to the preferred embodiment of the invention. The process begins at block **401** and thereafter proceeds to block **403**, which indicates application requesting specific memory allocation and the request being past to the OS. Following, the OS locks down a block of memory equal to the size of memory required by the application as shown at block **405**. The memory block is allocated and locked down in a single step utilizing the AWE APIs provided by the OS. The OS then forwards the virtual memory addresses of the memory that has been locked down to the kernel mode driver, which translates the virtual memory addresses into their associated physical addresses and returns the physical addresses to the OS as shown at block **407**. In the preferred embodiment, the OS, directed by the MAS, utilizes the standard AWE API calls to accomplish this task. Then, as shown at block **409**, MAS completes a check to determine if the physical memory that is locked down fits within the desired range pre-selected by the application developer. The pre-selected range is a parameter that is stored within MAS or the application and provided to the checking process when required. If the check indicates that there is a memory address within the locked down memory blocks that fits within/on the desired range, then another determination is made at block **411**, whether all of the requested specific memory has been locked down. If all the memory has been locked down, then MAS begins a process of deallocating memory blocks not within the range as shown at block **413**. Otherwise, the lock-step process (i.e., memory lock down, address translation, MAS checking for addresses within range) is repeated until physical memory runs out (or the maximum memory to allocate is reached) as indicated at block **415**.

During deallocation of memory, all or some of the memory not in the range that was locked down is deallocated and only the memory addresses that are in the range are returned (i.e., allocated) to the application process. In the preferred embodiment, while memory lock down is being repeated, the memory that was allocated (locked down) including memory determined to not be within the desired range is not released. Thus, only if physical memory runs out before the MAS process ends is the memory not within the range deallocated. When this occurs, the memory allocated within the range is passed back and additional memory may be allocated by the OS to supplement the memory still required by the application.

According to the preferred embodiment, deallocation of unwanted pages only occurs when all of the desired memory is already locked. This is because, if the memory was deallocated immediately after it was deemed to not fit in the range, then there is a likelihood that subsequent lock down processes may obtain the same memory page when another allocation is completed. After lock down process has been completed, the process ends as indicated at block 417.

There are several possible variations to the basic process described above. In a first embodiment, the OS determines if the desired amount of memory and range can be accommodated without having to necessarily keep grabbing more memory until the system is out of memory. Accordingly, possible values for the maximum amount of memory to allocate can be represented by one of several formulas. The maximum memory allocation may be calculated as being equal to: (1) amount of physical memory in the system (implemented in the basic algorithm); (2) (amount of physical memory in the system - size of the range asked for) + the amount of physical memory asked for; (3) amount of available physical memory in the system; or (4) (amount of available physical memory in the system - size of the range asked for) + the amount of memory asked for.

In another embodiment, instead of allocating physical memory in chunks that the programmer requested, memory is allocated all in one chunk based on the maximum amount of physical memory needed to get the desired range. In yet another embodiment, the physical memory that the user requested is allocated in chunks, but the chunk size is reduced as more memory is grabbed that is within the desired range.

Figure 5 is a flow chart of the process by which the invention is implemented, including many of the steps of **Figure 4**, but also illustrating a few other functional steps utilized in the organization and processing of the physical pages allocated. Thus, to the extent that functional blocks of **Figure 5** overlap in content with **Figure 4**, these blocks are not described in the description of **Figure 5**. Beginning at block **415**, the deallocation is completed via an ascending/descending sort as indicated at block **518**. All of the pages that fit within the range are sorted in ascending order, providing a continuous organization of the memory blocks. Following, a maximum continuity is found as indicated at block **520**. The MAS processing goes through all of the pages that were sorted and finds the section with the greatest sequential continuity. For example, there may be 200MB of pages that fit into the range, but the application developer may have only requested 100MB allocated to the application processes. Thus, the MAS checks the 200MB of memory looking for a continuous or almost continuous section with 100MB. This check may comprise moving a 100MB "window" through the 200MB of memory to determine, which position of the window yields the highest continuity. Following, as shown at block **522**, the memory blocks with the maximum continuity are passed to the application along with additional memory blocks if required.

Alternatively, the method of sorting though the allocated memory blocks is completed with a Window-like processing. When Windows OS allocates AWE physical pages, there is a certain pattern to the arrangement of the physical pages. Since the basic algorithm does not provide this pattern, an additional routine is coded within MAS.dll

to organize the physical pages in a manner, which keeps in the spirit of the Windows-like pattern. This organization/arrangement helps to make physical address arrangement appear to have been allocated (and thus arranged) by Windows OS, while still maintaining the desired range.

Further, the sorting and Windows-like routines may also be adjusted in various ways for time concerns. That is, the Windows-like routine could limit its search for a good Windows-like match, and the sorting routine could be set to "mostly" sort the pages instead of completely sorting the pages.

In yet another embodiment, an X separation is provided between the pages to create a unique effect. For example, a 32MB separation between pages could be helpful if a system has a 32MB cache and the processing is designed to focus on cache misses.

Figures 3A, 3B, and 3C provide descriptions of the implementation of the features of the invention in a multi-node data processing system. With a multi-node application, MAS has to provide consideration for memory allocation across nodes. The various implementation steps are similar to those in the single node system as described above. However, as those skilled in the art would be aware, the multi-node configuration offers specific implementation considerations that are different from the single node configuration. Turning now to **Figure 3A**, there is illustrated a multi-node configured data processing system. Data processing system **300** comprises first processing system **301A** and second processing system **301B**, which are each independent processing systems, similar to the data processing system of **Figure 1**. Thus, both first and second processing systems **301A** and **301B** comprises major components, including CPU **303A** and I/O devices **305A**. More importantly to the invention, however, each processing system comprises a separate memory **307A** and **307B** and thus separate memory controllers **308A** and **308B**.

Although second processing system **301B** is illustrated having similar components as first processing system **301A** (i.e., similarly configured), it is possible for second processing system **301B** to be configured very differently than first processing system **301A**. Both system may operate totally independent of each other and may occasionally operate in concert with each other when resources need to be shared. Connection between both systems is implemented via a node **311**, which may be a simple direct system bus connection, a switch, a network connection, or the like.

In the preferred embodiment, memory resources may be shared among first and second system **301A**, **301B**. Thus, memory controller **308A** may allocate/deallocate memory blocks in memory **307B**, for a process running on CPU **303A**. Likewise, memory controller **308B** may allocate/deallocate memory blocks in memory **307A**, for a process running on CPU **303B**.

Although, the illustrated embodiment depicts only basic components of a multi-node data processing system interconnected via a node, it is understood that data processing system **300** may be more complex. Data processing system **300** may comprise additional processing systems connected via additional nodes, with each processing system having additional components (or similar components with a different configuration).

Referring now to **Figure 3B**, there are illustrated software and hardware components of a multi-node data processing system illustrating the separate memory blocks and corresponding addresses. The alternate embodiment of the invention allows the specific allocation of the entire array of available memory, including memory situated across nodes (e.g., memory **307A** and memory **307B** of **Figure 3A**). **Figure 3B** illustrates the two memory blocks, memory **307A**, **307B** of **Figure 3A** and associated memory addresses. As seen by memory controller **308A**, memory **307B** is comprised of

5 tagged addresses, representing a continuation of the memory addresses available in memory 307A. As seen from their respective processors and memory controllers, both memory 307A and memory 307B includes address range 00000-31999. However, for simplicity, it is assumed herein that memory controller 308A sees memory 307B as having address range 32000-63999. Of course, the exact nature of sharing memory across nodes may vary such that specifying an address range may not require the controller viewing the separate memory as a contiguous memory block.

10 The invention is mainly concerned with allocating physical memory for software. However, the MAS.dll may be included within other tools implemented in software applications and may be extended to various other subsystems besides memory in the Windows 2000 environment. One major advantage of using this invention is that a user can lock down specific physical memory locations in a particular range of physical addresses for software. Additionally, the invention finds particular applicability to testing procedures for memory and memory operation, since the tester (designer of the test application) is able to isolate specific areas/blocks/addresses of memory for testing.

15 20 25 As a final matter, it is important that while an illustrative embodiment of the present invention has been, and will continue to be, described in the context of a fully functional data processing system, those skilled in the art will appreciate that the software aspects of an illustrative embodiment of the present invention are capable of being distributed as a program product in a variety of forms, and that an illustrative embodiment of the present invention applies equally regardless of the particular type of signal bearing medium used to actually carry out the distribution. Examples of signal bearing media include recordable media such as floppy disks, hard disk drives, CD-ROMs, and transmission media such as digital and analog communication links.

5